Slacking Off II

DiPS CodeJam 24

Prompt

Bobby, having having figured out how many programs he can compile, sits down and starts compiling. Bobby now sees a problem – in his quest for longer compile times, he's made a small error (I'll let you figure this one out yourself):

```
class X: Y
class Y: X
```

So he makes another list – this time denoting which classes are subclasses of which other classes. Can you find out if such a 'transitive equality' of **two** classes is present in the code?

Hint: this can also be expressed as a graph theory problem – *Given an undirected graph and a vertex in the graph, find the number of vertices in its connected component.*

Input Format

- The first line of the input contains an integer n, denoting the number of lists.
- The next n lines of the input each contain a space separated list of values conforming to $X = Y$, where X and Y denote class names.

Output Format

The first and only line of your output must contain a single integer m, denoting the number of lists where a transitive equality exists.

Constraints

- $10^2 \leq n \leq 10^3$
- The size of each list varies between 10^2 and 10^3 elements.

Sample Program

```
# lists = [  [("abc", "def"), ("ghi", "jkl"), ("mno, "pqr""), ...], ...  ]

def solve(lists):
  count = 0

  for l in lists:
    d = dict(l)
    keys = d.keys()
    for k in keys:
      v = d[k]
      if v in keys and d[v] == k:
        count+=1
```

```
return count
```